National Conference on Engineering Innovations in Emerging Technologies
In association with International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET219411

A Discriminative Model to Generate Melodies through Evolving LSTM Recurrent Neural Networks

Dr. Nanda Ashwin[1], Uday Kumar Adusumilli[2], Lakshmi Kurra[3], Prof. Kemparaju N[4]
[1]Professor, Dept. of Information Science and Engineering, East Point College of Engineering and Technology, Bangalore, India
nandaashwin@epcet.ac.in[1]
[2]Product Support Analyst, Associate, Infor, Bangalore, India
uday.adusumilli@infor.com[1]
[3]Student, Dept. of Information Science and Engineering, East Point College of Engineering and Technology, Bangalore, India
[4]Head, Dept. of Information Science and Engineering, East Point College of Engineering and Technology, Bangalore, India
hod.ise.epcet@eastpoint.ac.in[4]

ABSTRACT

The paper describes a method that uses evolving LSTM recurrent neural networks to generate melodic music through a discriminative model. The approach enclosed has achieved an accuracy level of over 90%, thus enabling our model to understand & generate music as per the input parameters. The input expected from the user is minimal and can be provided by a layman. The experiments presented here demonstrate how LSTM can successfully learn a form of training music data and compose a novel (and pleasing) melody based on that style of training. LSTM can play melodies with good timing and appropriate structure if the parameters have been set appropriately. The RNN Model presented in this paper leverages the benefits of LSTM networks and demonstrates how this feat can be achieved.

Keywords : LSTM, Music Generation, Deep Learning, Recurrent Neural Networks, RNN, Discriminative Model, Evolving LSTM, Evolving LSTM RNN

I. INTRODUCTION

In order to use an RNN as a single-step predictor, it is easiest to use the network to automatically compose music. It is designed to be able to predict notes at time t + 1 by using notes at time t as input and notes at time t as output. Using its own outputs to create subsequent inputs, the network can generate new compositions after the learning has stopped. It would be difficult for a feed-forward network to compose music this way. The inability of this network to store past data would mean that these networks would not have the ability to track where they are (or were) in a song. The limitation mentioned above should not apply to RNN models. In the brain, recurrent connections allow hidden layer activations to be used

Copyright: © the author(s), publisher and licensee Technoscience Academy. This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License, which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited

as memory. Thus it can exhibit (albeit arbitrary) temporal dynamics as a result of the hidden layer activations.

When faced with this task, however, RNNs typically fall short of expectations. It is due to the vanishing gradient property of RNNs that this failure occurred. As the error flow disappears or explodes exponentially in gradient methods, such as Real-Time Recurrent Learning (RTRL) and Back-Propagation Through Time (BPTT), the networks will have trouble processing long-term dependencies. In music, dependencies across multiple notes and beyond are critical for defining a certain style. This way, metrical structure and phrasal structure are formed over time. Changes in chords are a good example of this type of dependency. When a chord is held for four bars or longer, like with early rock-and-roll for instance, then this is a musical form. It is expected that networks spanning 32 events or more will be able to bridge regular and reliable periods or spans of notes smaller than eight. It is important to note that Mozer (1994) composed note-by-note melodies accompanying chord progressions of single-voice melodies.

The RNN techniques used in "CONCERT" included BPTT (a combination of statistical methods) and proportional likelihood methods. A psychologically realistic approach to input encoding has also been employed by the researchers (Shepard, 1982) alongside the neural network approach. As a result of a note that was related chromatically and harmonically, the network became inductive. The second encoding method used to produce distributed representations of chords was used in order to achieve the desired result. In our previous report, we discussed how BPTT-trained RNNs are ineffective at learning long-term dependencies. As a result, duration had to be encoded on a distributed basis, allowing him to process any note in a time interval on the network at the same time. Since single time steps are represented rather than slices of time in notes, the network has to bridge significantly fewer time steps while learning global structure. There is a network that encodes slices of time in an undeviating manner. 16th notes, for instance, require that the whole note span sixteen steps within the network.

Considering the sixteenth notes found in every CONCERT composition, every composition would require 172-time steps. CONCERT architecture's method of representing melody, chords, and duration in a psychologically realistic way did not accurately capture global musical structure despite being based on RNNs. Although networks regularly overproduce transition tables of third order, they did not find global structure in any case. The importance of structural multi-level structure is emphasized by Mozer in analyzing the performance of the note-by-note method. The following architectures may provide solutions, and we recommend them.

II. LSTM MUSIC COMPILER

A. LSTM Architecture

LSTM can't be described in depth due to space constraints. The concept of LSTMs is to maintain a constant flow of errors over time. A constant error carousel (CEC) is used by LSTMs to overcome the problems associated with error decay that plague previous RNNs to preserve the error flow from undesirable perturbations. An external cloud of nonlinear units is connected to a fixed self-configuration of CECs by means of a fixed self-configuration. Controlled data flow between nonlinear circuits is controlled by CEC. Flows that use multiplicative inputs become resistant to perturbation due to their multiplicative nature. Multiplicative output gates can also learn not to disturb other units if they receive memories that are currently irrelevant. It does this by resetting the contents of an outdated memory cell. An optimized version of BPTT was used in conjunction with a specialized version of BPTT for this RTRL. The input gates, the forget gates, and the weights to the cells use the RTRL implementation, and the output gates use a

pruned version of BPTT, in order to minimize the computation times. LSTMs perform better in online knowledge situations when weight updates are controlled by a Kalman filter.

B. Data Representation

Our goal is to avoid realistically distributed encodings and instead present simple descriptive descriptions of the data. Each note is assigned 1.0 input/target, with 0.0 representing off. Instead of CONCERT, this scheme relies on a network to learn to prioritize melodies that are harmonically and chromatically appropriate. Our preference for local representation is based on several reasons. Chords and melodies are not artificially distinguished, and there is an implicit sense of many voices. Actually, chords are implemented by enabling the right notes in an input vector. Additionally, it is simple to create probabilistic distributions over the possibilities, allowing the likelihood of a single note to be independent or dependent. A single input vector represents one slice of real time, with one denotation of time. Depending on the step size of the quantization (which is the case for all experiments in this study), a whole note can be processed in eight network time steps. Since this method involves learning the relative durations of notes, the algorithm is better suited for LSTM since it makes the mechanism of counting and timing simpler.

Despite the fact that this graphic is a representation of a complex issue, it misses two crucial points. A note cannot be stated precisely as to when it ends, but it is important that the listener keeps this in mind. The same is true for two eighth notes played at the same pitch. Thus, four sixteenth notes played at the same pitch will be expressed as eight eighth notes. By decreasing quantization steps and marking all note ends with zeros, altering input and target structures is not required. The exact same pitch is analyzed using a quantization level of 1 for every eighth note and four quarter notes assigned with the exact same quantization level. To indicate the beginning of a note, a unit (or units) can be created within the network. A method such as this is certainly feasible.

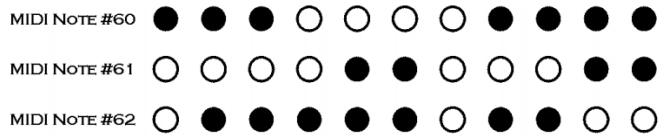
However, it is unclear how the same approach would work with data sets containing multi-voice melodies. Simulation results indicated that a range of 12 chords and 13 melodies is possible. In the simulations, we divided chord notes from melody notes and this separation is artificial since the chord notes and melody notes are not represented in a different manner. We also intend to integrate the two in a more realistic way in future experiments.

Our goal is to use MRF modeling within the polyphonic music domain. The representation of music can take many forms. A .wav or .mp3 file is a sequence of audio signals, in its most unstructured form. It is also possible for music to be represented as instructions for the performer, as in opus . The music represented in this way includes all the notes within a piece, as well as the onsets, durations (e.g., "quarter" notes, half notes"), and pitches of all the notes. In addition to the time signatures and key signatures, sharps, flats, ties, and slurs, this music provides other dynamic markings which help to instruct the performer on how they should perform the piece. There is a form we influence that is neither of these forms, but one that mixes aspects of the two forms. It is also possible to use MIDI (www.midi.org) as an example, even if the music does not have to be in this format. A MIDI file contains the start, duration, and pitch of every note in the piece of music. It is not always possible to obtain other information. The pitches are encoded as numbers from 1 to 128. Due to the lack of symbolic durations, milliseconds are the only units specified. There are no symbolic onset times, but rather millisecond integer locations.

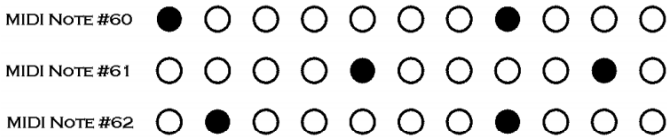
The concept of monophonic music is that in the event of a note playing, no new note may start until the previous note has been completed. Such a restriction does not exist in polyphonic music. The end of one note may be followed by the beginning of another. Thus, polyphonic MIDI music is represented visually

by a two-dimensional graph, on which millisecond time and the number of notes (1 to 128) are plotted.

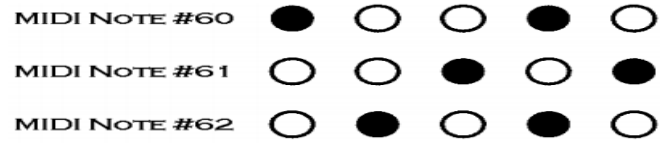
Here is a picture to illustrate this point. A black circle represents a note that is "on". An "off" note is represented by a white circle.



The purpose of our MRF modeling is to use the polyphonic content in our source documents to model. For each note in the sequence, only the onset times are selected, and the note's duration is ignored. Thus, the example above becomes:

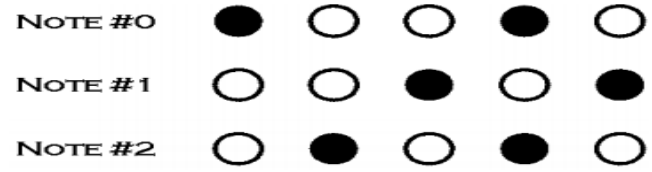


Our next step is to remove all millisecond onsets without any pitches. (A first-stage model must be able to discard not only the duration of notes, but also the intervals between notes.) Millisecond onset times with pitches, however, are called "simulation".



The 128-note y-axis has now been reduced to the equivalent 12-note octave pitch set. This is achieved by multiplying MIDI pitch numbers by the modulation-12 value.

The example above thus becomes:



There are either 1s or 0s in the sequential sequence of 12-element bit vectors; thus, there are 12 elements. For each spot, we counted whether a note had an onset at that particular time (mod 12). Throughout this paper, we do want to emphasize that it is not necessary that the source music be MIDI. We are mostly concerned about the pitch values and the relative order of notes. In addition to MIDI files, it could also have been derived from transcribed audio files (AMR) or scanned musical compositions (OMR). If pitch information is present in a piece of music, our algorithms will work on it.

III. EXPERIMENTS

LSTMs are demonstrated to be capable of learning to recreate a musical chord structure in this experiment. Rather than relying on melody for the chord structure, we want to ensure that it is possible to infer a chord structure in the absence of melody. If LSTMs are predicting global chord structure, they may benefit from the local structure in melody. Input data is especially at risk when recombination occurs using relatively few musical examples as in this case.

A. Restricted Boltzmann Machines (RBM)

Based on an energy model, the probabilities of the visible input vector v (inputs) and the hidden vector h for any given configuration of the vectors are:

$$P(v,h) = \exp(-b_v^{\mathrm{T}} v - b_h^{\mathrm{T}} h - h^{\mathrm{T}} W v)/Z \qquad (1)$$

b_v, b_h and W represent the parameters of the model, while Z represents the partition function. The hidden units h_i are conditionally independent of one another when the vector v is given, and the reverse is true when the vector v is given:

$$P(h_i = 1|v) = \sigma(b_h + Wv)_i \qquad (2)$$

$$P(v_j = 1|h) = \sigma(b_v + W^{\mathrm{T}} h)_j \qquad (3)$$

The element-wise logistic sigmoid function is $\sigma(x) \equiv (1 + e^{-x})^{-1}$ are calculated from the marginalized probability of $F(v)$ by $P(v) \equiv e^{-F(v)}/Z$:

$$F(v) = -b_v^{\mathrm{T}} v - \sum_i \log(1 + e^{b_h + Wv})_i \qquad (4)$$

The RBM inference process consists of sampling h_i given v (or the v_j given h), depending on the conditional distribution of h_i to the Distribution Bernoulli (eq. 2). Using block Gibbs sampling, i.e. alternating between sampling v and h in k steps, it can be efficiently sampled from the RBM. It involves

two opposing terms, the positive phase, and the negative phase, of an input vector $v^{(l)}$.

B. The RNN-RBM

Recurrent Temporal RBMs (RTRBM) can be viewed as sequences of conditional RBMs. It works by describing the conditional distributions and conveying temporal information in hidden units, which are outputs of a deterministic RNN. An RNN with distinct hidden units $h^{\wedge(t)}$ can be combined to alleviate this constraint with the RTRBM graphical model as shown in Figure 1. We call this model the RNN-RBM.

For simplicity, we consider the RBM parameters to be W, $b_v^{(t)}$, $b_h^{(t)}$ (i.e. only the biases are variable) and a single-layer RNN (bottom portion of Fig. 1(b)) whose hidden units $h^{\wedge(t)}$ are only connected to their direct predecessor $h^{\wedge(t-1)}$ and to $v^{(t)}$ by the relation.

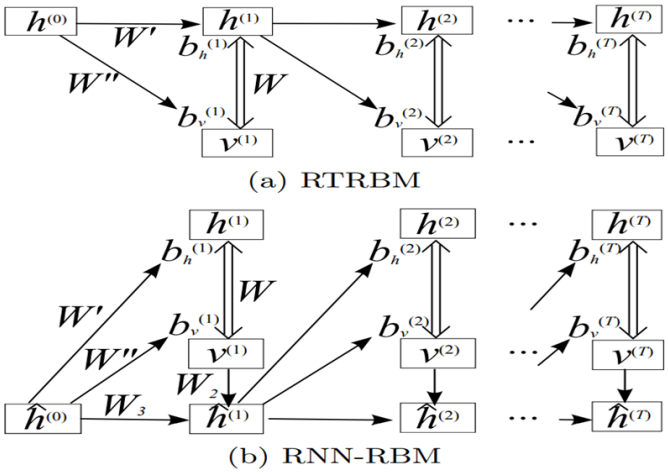


Figure 1: Graphical structure comparison of (a) the RTRBM and (b) the RNN-RBM with only one layer. An RBM's single-arrow representation represents its deterministic functioning, while its two-arrow representation represents its stochastic hidden-visible connections. An RNN with $h^{\wedge(t)}$ underlying units is displayed on the RNN-RBM's upper half. A linear relationship exists between RBM biases $b_h^{(t)}$, $b_v^{(t)}$ in relation to $h^{\wedge(t-1)}$.

C. Resilient Propagation (RProp)

A heuristic optimization algorithm called resilient propagation (RProp) is used for training the long-short-term memory (LSTM) network. It is a local adaptation learning strategy that modulates weight-specific parameters, such as learning rate, in order to facilitate gradient-descent learning problems. It is possible to achieve this result by only utilising weight-specific information, in this case partial derivatives, which are available in this case. By ignoring the scale of the partial derivatives, RProp directly adapts the weight update's dimensions. In contrast to other adaptive learning algorithms that use the gradient magnitude to adjust learning, in this case, only the derivative's sign is used. Load-related updates are determined solely by weight-specific update values, so the sign of the derivative tells us the direction of the load update.

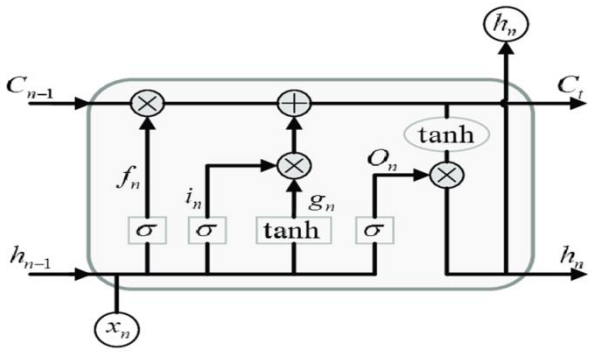


Figure 3: The Structure of Long-Short Term Memory Neurons

The RProp algorithm has been found to achieve exponential convergence faster than plain gradient descent algorithms in research studies (Riedmiller, 1994). Furthermore, we do not have to tune the parameters for optimal results with RProp.

D. Melody-RNN

The Melody-RNN library is an open source project built by Google under its Magenta operating system. We believe one of Magenta's primary objectives is to make music and art production more intelligent with machine intelligence using a broad array of machine learning techniques. There has already been extensive use of machine learning in understanding content, mostly in recognition of speech or translation of text. Through the use of intelligent algorithms and systems,

Project Magenta explores the other side, developing algorithms that can learn to create music and art on their own, therefore creating the potential for compelling and artistic content. LSTM networks can be comprised of two layers and Melody RNNs are designed to be simple twin-layer LSTM networks. Presently, Melody-RNN models can be categorized into three groups. In the first model, we use basic one-hot encoding to represent melodies as inputs to the LSTM; in the second one, we use Lookback RNN, which is a model that introduces custom inputs and labels to allow the model to easily recognize pattern across one and two bars; the other one is Attention RNN, which applies attention to the RNN cell to let it access information from the past without storing it in its state.

Magenta team has recently published an update showing that the DQN network can be applied to the Magenta generation process to work as a reward function to train the neural network to follow certain theories of music. Below is a summary of the basic concept:

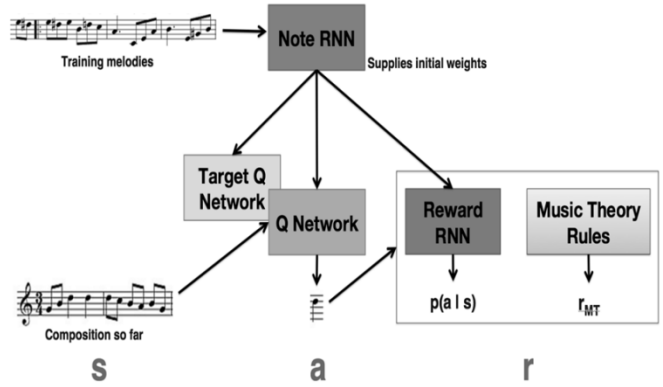


Figure 4: WaveNet Network Architecture

Attention RNN is by far the most sophisticated and best performing model compared to the others. Using a RNN layer with a size of 128x128 and a batch size of 32, the dropout rate was set to 0.5, and the initial learning rate was 0.01.

E. Biaxial-RNN

One of the impressive takeaways of Daniel Johnson's impressive RNN music composing project is the Biaxial-RNN. The following properties make it well designed:

1. Comprehend time signatures: compositions must adhere strictly to time signatures.
2. Consistency across time and space: ability to compose indefinitely.
3. Having the same structure of the network as you transpose the notes up and down.
4. Allow the selection of coherent chords and the simultaneous playing of multiple notes.
5. Permit repeating a note: holding the C note for two beats is different from playing it twice.

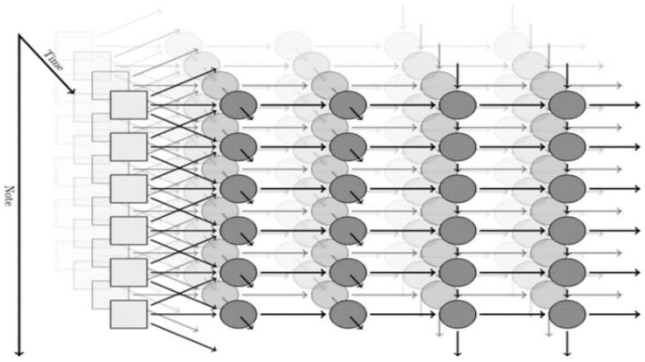


Figure 5: Biaxial-RNN Network Architecture

Several existing RNN-based compositional techniques have invariants with respect to notes but variable behaviors with respect to time. Imagine if we transpose one octave up and were expecting the model to produce a piece of music that is almost identical to the original. By doing this, the Biaxial-RNN design passes down history information along two axes (the note axis and the time axis). As inputs to the model, pitch class, proximity, note value, past context, and beat information; as outputs, articulate probability and play probability are taken into account.

F. Sample Training Input to Model

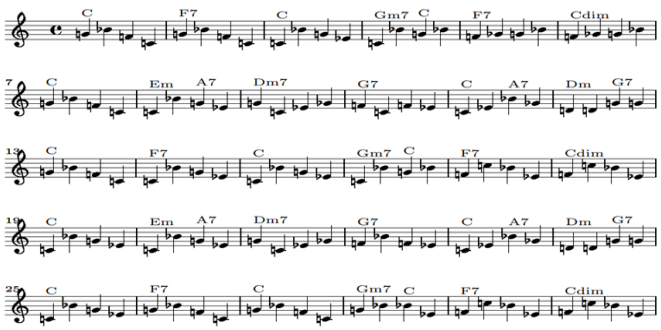


Figure 2: Sample representation of the input music notes provided as Training Data

IV. CONCLUSION

As demonstrated in this paper, discriminative neuroevolution can be used to create artificial compositions that draw from underlying rules and principles of music theory. Through a combination of common composition rules and specific style descriptions, this program has demonstrated the capability of composing melodies that are similar to the music desired. Despite being relatively simple, the results show promise, and it is likely that they will improve with the addition of more composition rules and larger training datasets.

V. REFERENCES

[1]. Breuleux, O., Bengio, Y., and Vincent, P. (2011). Quickly generating representative samples from an RBM-derived process. Neural Computation, 23 (8), 20532073.

[2]. Karpathy, A., The unreasonable effectiveness of recurrent neural networks, 2015.

[3]. Eck, D. and Schmidhuber, J. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In NNSP, pp. 747–756, 2002.

[4]. D. A. and M. Clements. Melody spotting using hidden markov models. In J. S. Downie and D. Bainbridge, editors, Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR), pages 109–117, Indiana University, Bloomington, Indiana, October 2001.

[5]. J. P. Bello, L. Daudet, and M. B. Sandler. Time-domain polyphonic transcription using self-generating databases. In Proceedings of the 112th Convention of the Audio Engineering Society, Munich, Germany, May 2002.

[6]. D. Temperley, A probabilistic model of music perception, in: Proceedings of the 7th Conference on Music Information Retrieval, Victoria, Canada, 2006.

[7]. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K., 2016. WaveNet: A generative model for raw audio.arXiv preprint arXiv:1609.03499.

[8]. Daniel Johnson, Composing Music With Recurrent Neural Networks, 2015

[9]. Theis, L., Oord, A.V.D. and Bethge, M., 2015. A note on the evaluation of generative models. arXiv :1511.01844.

[10]. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets In Advances in Neural Information Processing Systems (pp. 2672-2680).

[11]. Eck, D., Schmidhuber, J. (2002) A First Look at Music Composition using LSTM Recurrent Neural Networks. Technical Report No. IDSIA-07-02

[12]. Sutskever, I., Hinton, G., and Taylor, G. The recurrent temporal restricted Boltzmann machine. In NIPS 20, pp. 1601–1608, 2008.

[13]. Antokoletz, E. (1981). The Musical Language of Bartok's 14 Bagatelles for Piano. Tempo, Vol. 137.

[14]. Allen Huang and Raymond Wu. Deep learning for music, June 2016. arXiv:1606.04930v1.

[15]. Chun-Chi J. Chen and Risto Miikkulainen. Creating melodies with evolving recurrent neural networks. Proceedings of the 2001

International Joint Conference on Neural Networks, 2001.

[16]. J.-F. Paiement, D. Eck, S. Bengio, Probabilistic melodic harmonization, in: Proceedings of the 19th Canadian Conference on Artificial Intelligence, Springer, 2006.

[17]. V. Lavrenko, J. Pickens, Polyphonic music modeling with random fields, in: Proceedings of ACM Multimedia, Berkeley, CA, 2003.

[18]. J. P. Bello, J. Pickens, A robust mid-level representation for harmonic content in music signals, in: Proceedings of the Sixth International Conference on Music Information Retrieval, London, 2005.

[19]. Franklin, Judy A. Recurrent neural networks for music computation. INFORMS Journal on Computing, 18(3):321–338, 2006.

[20]. Godbole, Shantanu and Sarawagi, Sunita. Discriminative methods for multi-labeled classification. In Advances in Knowledge Discovery and Data Mining, pp. 22–30. Springer, 2004.

[21]. Mozer, Michael C. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. Connection Science, 6(2-3):247–280, 1994.

[22]. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods – Support Vector Learning, pages 185–208. MIT Press, Cambridge, MA, 1998.

[23]. Rossi, G. Girolami, and M. Leca. Identification of polyphonic piano signals. Acustica, 83:1077–1084, 1997.

[24]. Ryyn¨anen and A. Klapuri. Polyphonic music transcription using note event modeling. In Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, October 2005.

[25]. Schlkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. Neural Computation, 13(7):1443–1471, 2001. A.D. Sterian. Model-based Segmentation of Time-Frequency Images for Musical Transcription. PhD thesis, University of Michigan, 1999.

[26]. J.P. Bello, L. Daudet, and M. Sandler. Time-domain polyphonic transcription using self-generating databases. In Proc. 112th Convention of the Audio Engineering Society, Munich, May 2002.

[27]. A.T. Cemgil, H.J. Kappen, and D. Barber. A generative model for music transcription. To Appear in IEEE Transactions on Speech and Audio Processing, 2005.

[28]. S. Dixon. On the computer recognition of solo piano music. In Proc. Australasian Computer Music Conference, Brisbane, 2000.

[29]. D. Ellis and G. Poliner. Classification-based melody transcription. Machine Learning, 2006.

[30]. S. Godsill and M. Davy. Bayesian harmonic models for musical pitch estimation and analysis. In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Orlando, 2002.

[31]. K. Kashino and S. Godsill. Bayesian estimation of simultaneous musical notes based on frequency domain modelling. In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Montreal, 2004.